Programarea de sistem și de rețea

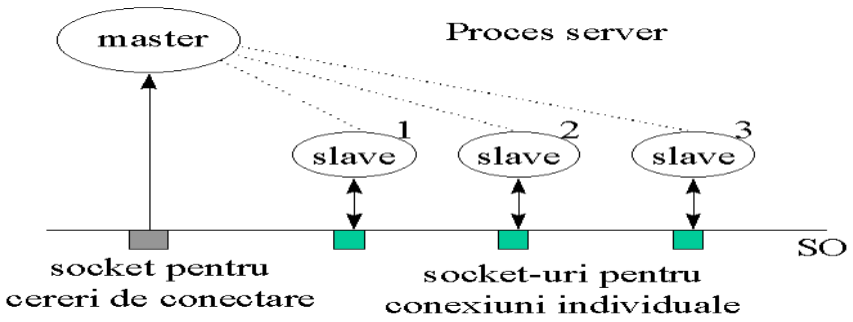
Lucrare de laborator nr.8

Proiectarea unui server web concurent

Lucrarea respectiva este o dezvoltare a lucrării precedente unde ați implementat un server web secvențial. Pentru a putea realiza sarcina acestei lucrări trebuie sa aveți codul sursă funcțional pentru lucrarea precedenta.

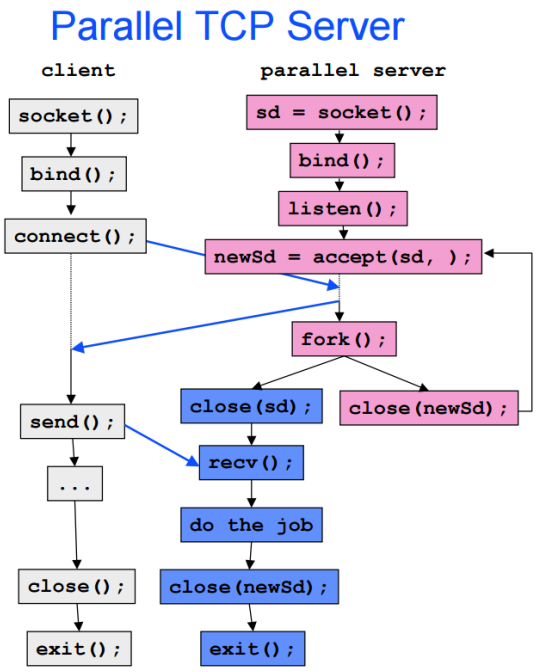
Probabil ca ați înțeles ca serverul proiectat în lucrarea precedentă este capabil sa servească simultan doar un singur client. Dacă se conectează clienți noi în timpul când serverul servește un alt client, el nu va răspunde la cerere decât după ce va termina servirea clientului actual. Se mai spune ca acest server este de tip iterativ. El funcționează bine doar in cazul când fiecare client necesită o durată scurtă pentru a fi servit. În cazul când durata de servire este lungă, durata de așteptare a celorlalți clienți poate deveni jenantă. Se dorește deci implementarea unui server care ar putea servi simultan mai mulți clienți.

In exercițiul ce urmează vom ameliora codul serverului precedent pentru a-l face capabil să gestioneze servicii concurente. Concurența unui server se obține folosind mai multe procese. Ideea este următoare: exista un proces server pricipal (master), care este executat inițial. Acesta deschide socket-ul la un port dat și așteaptă cererile. Pentru procesarea unei cereri acesta creează un proces server secundar (slave) care comunică direct cu clientul. După transmiterea răspunsului către client acest proces se va termina. Schema de principiu a unei astfel de comunicări este ilustrată în figura de mai jos.



Programarea concurenta poate fi realizata prin implementarea firelor de execuție (*thread-uri),* această tehnică ați studiat+o în cursul de programare concurentă și ea poate fi aplicată pentru orice sistem de operare. In sistemul de operare Unix e posibila de asemenea utilizarea apelului de sistem *fork()* prin care un proces-părinte poate crea procese-copii. În această lucrare ne vom axa pe cea de-a doua metodă, ea fiind mai simpla pentru că utilizează direct un apel sistem și nu necesită un efort suplimentar de programare.

Schema funcționarii acestui server e prezentată în continuare



Consultaţi [2] pentru detalii asupra posibilităţilor creării unui server concurent cu Unix. Studiaţi cu atenţie codul propus ca exemplu care realizează un server de ecou concurent şi identificaţi structurile principale de care va veţi servi pentru a crea un server web concurent. Prima etapă constă în a crea o buclă (`while`) cu instrucţiunea `accept()` cu scopul ca serverul sa poată gestiona clienţi noi. A doua ameliorare consistă în a utiliza un alt proces pentru a servi fiecare client conectat. Gestionarea se va efectua printr-o funcţie în `main`. După apelul `accept()` trebuie sa creaţi un nou proces cu ajutorului apelului `fork()`. Procesul copil va trebui sa gestioneze în continuare conexiunea (prelucrarea cererii, dialogul cu clientul, etc.). Procesul părinte va continua în buclă pe apelul `accept()` pentru a răspunde la eventualele conexiuni de intrare.

Notă : nu uitaţi sa închideţi conexiunea nouă înainte de a intra în buclă.

Puteţi consulta şi alte surse de pe Internet, citaţi în raport cele mai utile şi mai interesante dintre ele.

Testaţi funcţionarea serverului cu mai mulţi clienţi. Demonstraţi corectitudinea funcţionării lui în reţea. Vi se cere demonstrarea funcţionarii programului în sala de laborator utilizând reţeaua reală.

Conţinutul raportului

- Pagina cu titlu
- Scopul lucrării
- Descrierea codului
- Rezultatele experienţelor
- Concluzii

Depuneţi raportul şi o arhivă cu codul elaborat cel târziu după două săptămâni de la îndeplinirea lucrării.

Referinţe

1. Radu-Lucian Lupsa "Retele de calculatoare", capitolul 6 "Programarea în rețea – introducere", Casa Cărții de Știință, 2008
2. Michael J. Donahoo and Kenneth L. Calvert "TCP/IP Sockets in C. Practical Guide for Programmers", MKP, 2001
3. Ghidul Beej pentru programarea in rețea folosind socket de internet http://weknowyourdreams.com/beej.html